PROJECT PLANNING AND SCHEDULING

Planning and scheduling

- Levels of planning and scheduling
  - Long-range planning (several years),
  - middle-range planning (1-2 years),
  - short-range planning (few months),
  - **scheduling** (few weeks), and
  - reactive scheduling (now)
- These functions are now often integrated

Project planning

- Planning and scheduling of *jobs* (*activities*) subjected to precedence constraints.
- Setting is a parallel machine environment with unlimited number of machines.
- **Objective**: minimize makespan

Project properties

- Project goals: quality, time, costs, customer satisfaction
- Network of activities/jobs
- Limited resource capacity
- Project life-cycle:
  - Order acceptance
  - Engineering and process planning
  - Material and resource scheduling
  - Project execution
  - Evaluation and service

Project areas

- Construction
- Production
- Management
- Research
- Maintenance
- Installation, implementation
- **Examples**: construction of power generation centers, software developments, launching of aircrafts, design, development and construction of defense vehicles.

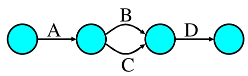
Project representation

- Example: planning a concert

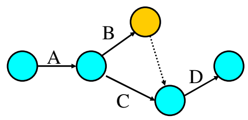
Job	Description	Predecessors
A	Plan concert	-
B	Advertise	A
C	Sell tickets	A
D	Hold concert	B, C

Job-on-arc format

- ❑ *Not allowed*: two jobs cannot have the same starting and ending node!



- ❑ Need to introduce a dummy job:



Example 1

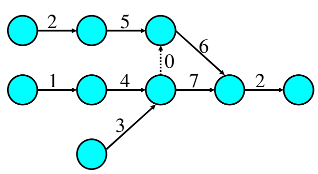
- ❑ Setting up a production facility

Job	Description	Duration (p_j)
1	Design production tooling	4 weeks
2	Prepare manufacturing drawings	6 weeks
3	Prepare production facility for new tools and parts	10 weeks
4	Procure tooling	12 weeks
5	Procure production parts	10 weeks
6	Kit parts	2 weeks
7	Install tools	4 weeks
8	Testing	2 weeks

Precedence graph

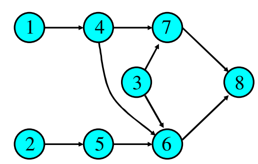
Job	Immediate predecessors	Immediate successors
1	-	4
2	-	5
3	-	6, 7
4	1	6, 7
5	2	6
6	3, 4, 5	8
7	3, 4	8
8	6, 7	-

Job-on-arc network



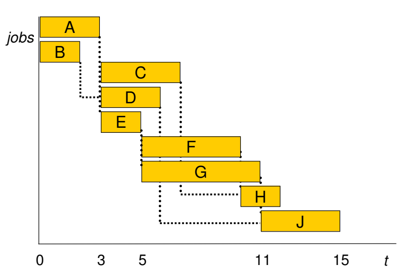
Job-on-node format

- ❑ No need for a dummy node
- ❑ Nodes can be depicted as rectangles



Jobs on a time axis

j	precedence
A:	-
B:	-
C:	A
D:	A, B
E:	A
F:	E
G:	E
H:	C, F
J:	D, G



Critical Path Method (CPM)

- ❑ Processing time, p_j, of job j is fixed. *A job does not require any resource*.
- ❑ Unlimited number of machines in parallel and n jobs with precedence constraints.
- ❑ **Objective**: minimize makespan
- ➢ **slack job**: the start of its processing time can be postponed without increasing the makespan
- ➢ **critical job**: the job that cannot be postponed
- ➢ **critical path**: the set of critical jobs

Critical Path Method

- **Forward procedure**
  - Starting at time zero, calculate the **earliest** each job can be started
  - The completion time of the last job is the makespan
- **Backward procedure**
  - Starting at time equal to the makespan, calculate the **latest** each job can be started so that the makespan obtained in the forward procedure is realized.
  - Finds the **critical path**.

Critical Path Method

Notation:

- p_j processing time of job j
- S'_j earliest possible starting time of job j
- C'_j earliest possible completion time of job j
- S''_j latest possible starting time of job j
- C''_j latest possible completion time of job j
- $C'_j = S'_j + p_j$
- $\{\text{all } k \to j\}$ jobs that are predecessors of job j
- $\{j \to \text{all } k\}$ jobs that are successors of job j

Forward procedure

Step 1:

Set time $t = 0$

Set $S_j' = 0$ and set $C_j' = p_j$. for all jobs j with no predecessors

Step 2:

Compute for each job j

$$S'_j = \max_{\{\text{all } k \to j\}} C'_k,$$

$C_j' = S_j' + p_j$

Step 3:

The optimal makespan is $C_{\max} = \max\left(C'_1, C'_2, \ldots, C'_n\right)$

Backward procedure

Step 1:

Set $t = C_{\max}$

Set $C_j'' = C_{\max}$ and $S_j'' = C_{\max} - p_j$ for jobs j with no successors

Step 2:

Compute for each job j

$$C''_j = \min_{\{j \to \text{all } k\}} S''_k,$$

$S_j'' = C_j'' - p_j$

Step 3:

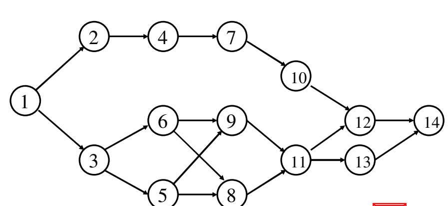
Verify that $\min\left(S''_1, \ldots, S''_n\right) = 0$

Comments

- The forward procedure gives the earliest possible starting time for each job
- The backwards procedures gives the latest possible starting time for each job
- If these are equal the job is a **critical job**.
- If these are different the job is a **slack job**, and the difference is the **float**.
- A **critical path** is a chain of jobs starting at time 0 and ending at $C_{\max}$.

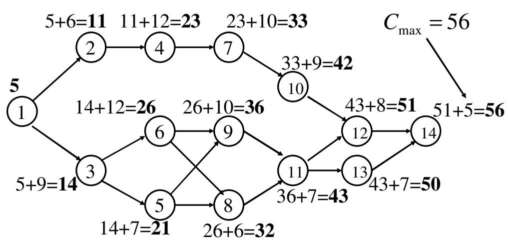
Example 2

Jobs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
p_j	5	6	9	12	7	12	10	6	10	9	7	8	7	5



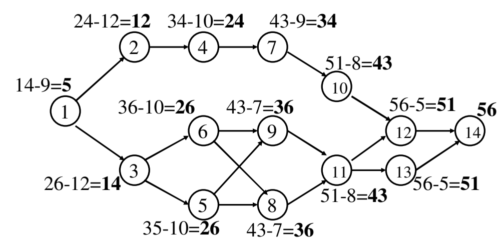
Precedence graph

Example 2: forward procedure



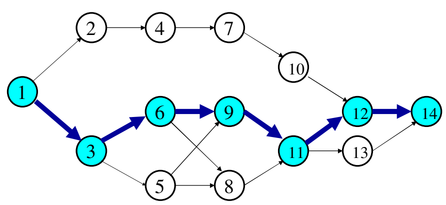
Jobs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
C'_j	5	11	14	23	21	26	33	32	36	42	43	51	50	56

Example 2: backward procedure



Jobs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
C''_j	5	12	14	24	30	26	34	36	36	43	43	51	51	56

Example 2: Critical Path



Program Evaluation and Review Technique

- Processing times is not deterministic in **PERT**.
- Processing time of job j is random with mean μ_j and variance σ_j^2.
- Want to determine the **expected makespan**
- Assuming:
  - p_j^a = optimistic processing time of job j
  - p_j^m = most likely processing time (mode) of job j
  - p_j^b = pessimistic processing time of job j

Expected Makespan

- Estimation of *expected processing time*:

$$\hat{\mu}_j = \frac{p_j^a + 4p_j^m + p_j^b}{6}$$

- Apply CPM with expected processing times.
- Let J_{cp} be a critical path.
- Estimation of **expected makespan**:

$$\hat{E}(C_{\max}) = \sum_{j \in J_{cp}} \hat{\mu}_j$$

Distribution of makespan

- Estimation of the variance of processing times:

$$\hat{\sigma}_j^2 = \left(\frac{p_j^b - p_j^a}{6}\right)^2$$

- Estimation of the variance of the makespan

$$\hat{V}(C_{\max}) = \sum_{j \in J_{cp}} \hat{\sigma}_j^2$$

- Assume it is normally distributed (Gaussian)

Example 2 using PERT (1)

Jobs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
p^a_j	4	4	8	10	6	12	4	5	10	7	6	6	7	2
p^m_j	5	6	8	11	7	12	11	6	10	8	7	8	7	5
p^b_j	6	8	14	18	8	12	12	7	10	15	8	10	7	8

- Precedence constraints as before.
- Estimation of means and standard deviations:

Jobs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
μ_j	5	6	9	12	7	12	10	6	10	9	7	8	7	5
σ_j	0.33	0.67	1	1.33	0.33	0	1.33	0.33	0	1.33	0.33	0.66	0	1
σ^2_j	0.11	0.44	1	1.78	0.11	0	1.78	0.11	0	1.78	0.11	0.44	0	1

Example 2 using PERT (2)

- Critical path is the same:

$$1 \to 3 \to 6 \to 9 \to 11 \to 12 \to 14$$

- Estimated makespan:

$$\hat{E}(C_{\max}) = \sum_{j \in J_{cp}} \hat{\mu}_j = 56$$

- Estimate of the variance of the makespan:

$$\hat{V}(C_{\max}) = \sum_{j \in J_{cp}} \sigma_j^2 = 2.66$$

Example 2 using PERT (3)

- Probability that the project is completed by time 60 is:

$$\Phi\left(\frac{60-56}{\sqrt{2.66}}\right) = \Phi(2.449) = 0.993$$

  - where $\Phi(x)$ is the probability that a normally distributed random variable ($\mu = 0$, $\sigma = 1$) is less than x.
- For the path: $1 \to 2 \to 4 \to 7 \to 10 \to 12 \to 14$ the estimated makespan is 55. The variance is 7.33
- Probability that the project is completed by time 60 is:

$$\Phi\left(\frac{60-55}{\sqrt{7.33}}\right) = \Phi(1.846) = 0.968$$

Potential problems with PERT

- Always underestimates project duration
  - other paths may delay the project
- Non-critical paths ignored
  - critical path probability
  - critical activity probability
- Activities are not always independent
  - same raw material, weather conditions, etc.
- Estimates may be inaccurate

Time/Cost tradeoffs: linear costs

- By **allocating money** (for additional resources) to jobs their processing time p_j can be reduced
- Assume that processing times are fixed
- Linear relation between allocated money and p_j
- Processing time:

$$p_j^{\min} \le p_j \le p_j^{\max}$$

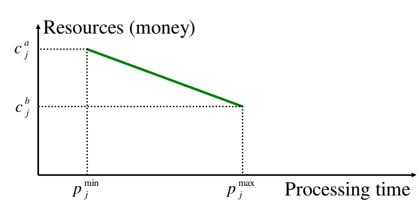
- Marginal cost of reducing processing time by one time unit:

$$c_j = \frac{c_j^a - c_j^b}{p_j^{\max} - p_j^{\min}}$$

Linear cost

- Cost of processing job j in p_j time units ($p_j^{\min} \le p_j \le p_j^{\max}$):

$$c_j^b + c_j\left(p_j^{\max} - p_j\right)$$

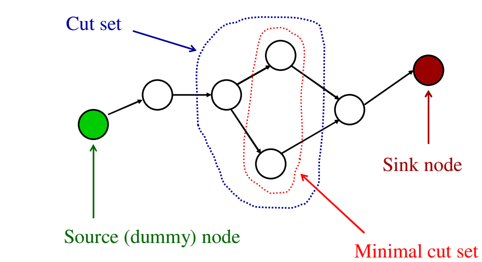


Solution methods

- **Objective**: minimum cost of project
- **Time/Cost Trade-Off Heuristic**
  - Good schedules (not optimal)
  - Works also for nonlinear costs
- **Linear programming formulation**
  - ***Always*** optimal schedules
  - Nonlinear version not easily solved

Sources, Sinks, and Cut sets

G_{cp} – subgraph consisting of critical paths



Time/Cost Trade-Off Heuristic

Step 1:
- Set all processing times at their maximum: $p_j = p_j^{\max}$
- Determine all critical paths with these processing times
- Construct the graph G_{cp} of critical paths

Step 2:
- Determine all minimum cut sets in the current G_{cp}
- Consider only those cut sets where all processing times are larger than their minimum: $$p_j > p_j^{\min}, \forall j \in G_{cp}$$
- If there is no such set STOP

Time/Cost Trade-Off Heuristic

Step 3:
- For each minimum cut set compute the cost of reducing all its processing times by one time unit.
- Take the minimum cut set with the lowest cost.
- If this is greater or equal than the overhead cost c_o per unit time STOP

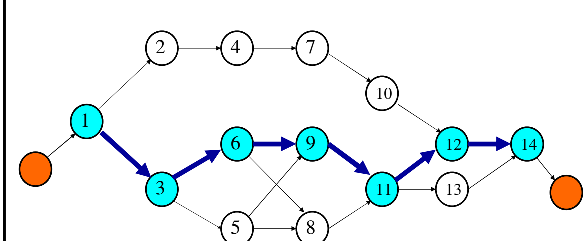
Step 4:
- Reduce all processing times in the minimum cut set by one time unit
- Determine the new set of critical paths
- Revise graph G_{cp} accordingly and go back to Step 2

Example 3 (example 2 modified)

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$p_j^{\max}$	5	6	9	12	7	12	10	6	10	9	7	8	7	5
$p_j^{\min}$	3	5	7	9	5	9	8	3	7	5	6	5	5	2
c_j^a	20	25	20	15	30	40	35	25	30	20	25	35	20	10
c_j	7	2	4	3	4	3	4	4	4	5	2	2	4	8

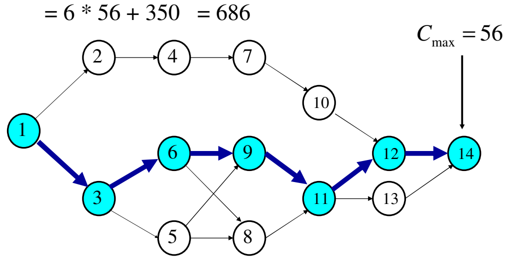
Overhead: $c_o = 6$ (cost of project per time unit)

1. Max. processing times, find G_{cp}

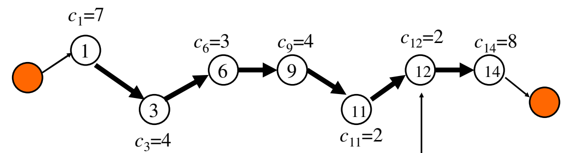


1. Max. processing times, find G_{cp}

- Cost = overhead + job costs = $c_o * C_{\max} + \Sigma c_j^a$
  $= 6 * 56 + 350 = 686$



2, 3. Min. cut sets in G_{cp} & lowest cost

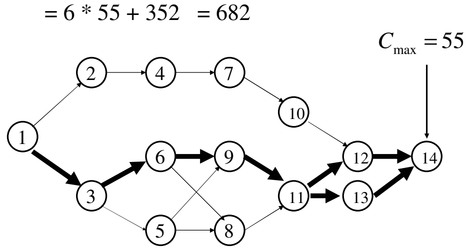


Cut sets: {1},{3},{6},{9}, {11},{12},{14}.

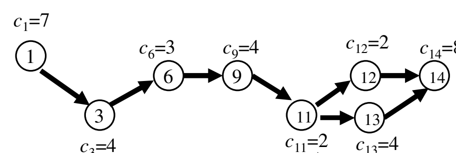
Minimum cut set with lowest cost

4. Reduce proc. time for each job by 1

- Cost = overhead + job costs = $c_o * C_{\max} + \Sigma c_j^a$
  $= 6 * 55 + 352 = 682$



2, 3. Min. cut sets in G_{cp} & lowest cost

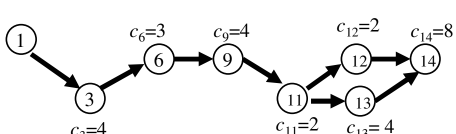


Cut sets: {1},{3},{6},{9}, {11},{12,13},{14}.

Minimum cut set with lowest cost

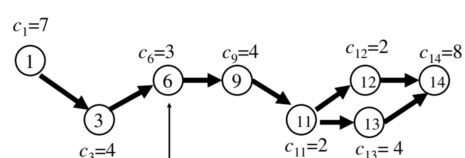
Next 3 iterations

- Cost = overhead + job costs = $c_o * C_{\max} + \Sigma c_j^a$
  $= 6 * 52 + 355 = 667$



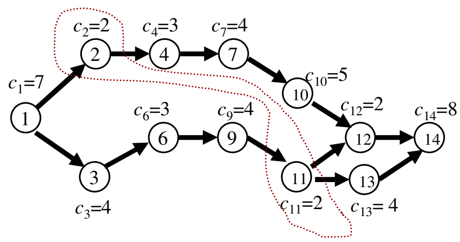
Next 3 iterations reduce processing time from 7 to 4

Step 1, 2, and 3

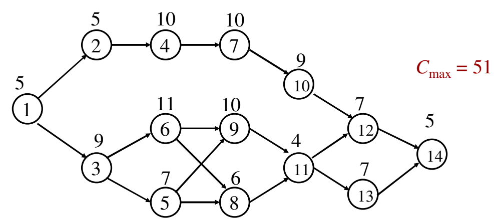


Reduce processing time next on job 6

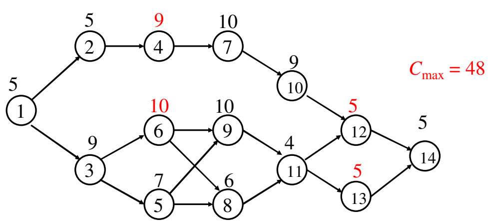
After more iterations …



Obtained solution



Other optimal solution



Linear programming formulation

- The heuristic does not guarantee optimum
  - See example 4.4.3 from Pinedo's book
- Here total cost is linear so use LP

$$c_o C_{\max} + \sum_{j=1}^{n} \left(c_j^b + c_j \left(p_j^{\max} - p_j \right) \right)$$

- Ignoring the constant terms:

$$c_o C_{\max} - \sum_{j=1}^{n} c_j p_j.$$

Linear Program

- The processing time p_j of a job j is a **decision variable**.
- The earliest possible starting time of a job j is denoted as x_j, and is also a decision variable.
- For jobs k without predecessors $x_k = 0$, and
  $p_j^{\min} \le p_j \le p_j^{\max}$
- The linear problem has $2n + 1$ decision variables: $C_{\max}, p_1, \ldots, p_n, x_1, \ldots, x_n$.

Linear Program

minimize

$$c_o C_{\max} - \sum_{j=1}^{n} c_j p_j.$$

subject to

$$x_k - p_j - x_j \ge 0, \forall j \to k \in A$$
$$p_j \le p_j^{\max}, \forall j$$
$$p_j \ge p_j^{\min}, \forall j$$
$$x_j \ge 0, \forall j$$
$$C_{\max} - x_j - p_j \ge 0, \forall j$$

Nonlinear costs

Discrete time-framework:

- Arbitrary function $c_j(p_j) \rightarrow$ cost of setting job j to processing time p_j
- decreasing convex cost-function

$$c_j(p_j - 1) - c_j(p_j) \geq c_j(p_j) - c_j(p_j + 1)$$

- non-decreasing overhead cost-function $c_o(t)$
- Given processing times and $c_j(p_j)$, which algorithm can be used (heuristic or LP)?

Nonlinear costs

Continuous time-framework:

- **Nonlinear programming** problem with the same constraints as the LP model.
- Objective function:

$$\int_0^{C_{\max}} c_o(t)\,dt + \sum_j c_j(p_j)$$

Workforce (Resource) constraints

- **Project Scheduling with Workforce Constraints** = Resource Constrained Project Scheduling (RCPS)

Notation

- n jobs j = 1,…, n
- N different pools in workforce i = 1,…, N
- W_i total number of operators in pool i
- W_{ij} number of operators in pool i needed for job j
- **Objective**: minimize makespan

Example 4

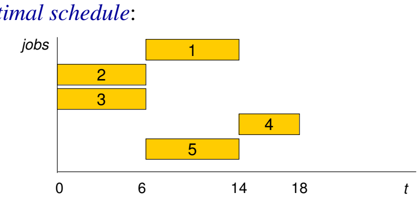
- Five jobs and two types of operators:
  - 4 operators of type 1
  - 8 operators of type 2

Jobs	1	2	3	4	5
p_j	8	4	6	4	4
W_{1j}	2	1	3	1	2
W_{2j}	3	0	4	0	3

Job	Immediate Predecessors	Immediate Successors
1	–	4
2	–	5
3	–	5
4	1	–
5	2, 3	–

Solution of example 4

- Without workforce constraints, critical path is 1 → 4, with $C_{\max}$ = 12.
- Optimal schedule: $C_{\max}$ = 18.
- *Optimal schedule*:



Workforce Constraints

- This is a **very hard** problem. Cannot be solved using linear programming.
- Can be formulated as an **integer programming** problem.
- Let job n + 1 be a dummy job (sink)
- Let x_{jt} be a binary variable:

$$x_{jt} = \begin{cases} 1 & \text{if job } j \text{ is completed exactly at time } t \\ 0 & \text{otherwise.} \end{cases}$$

Makespan

- Let H be an upper bound of makespan, as e.g.:

$$H = \sum_{j=1}^{n} p_j$$

- Completion time of job j is

$$\sum_{t=1}^{H} t\, x_{jt}$$

- Makespan:

$$\sum_{t=1}^{H} t\, x_{n+1,t}$$

Integer Programming formulation

minimize $\sum_{t=1}^{H} t\, x_{n+1,t}$

subject to $\sum_{t=1}^{H} t\, x_{jt} + p_k - \sum_{t=1}^{H} t\, x_{kt} \leq 0$ precedence constraints

$$\sum_{j=1}^{n} \left(W_{ij} \sum_{u=t}^{t+p_j-1} x_{ju} \right) \leq W_j \quad \text{total demand of pool } i$$

$$\sum_{t=1}^{H} x_{jt} = 1 \quad \text{jobs are processed}$$

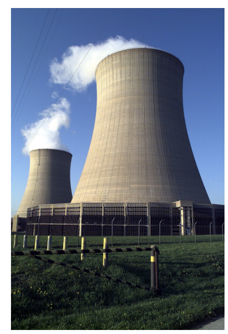
In practice

- The IP problem is very hard to be solved when number of jobs is large and time horizon is long.
- (Almost) always resource constraints
- ⇒ **Heuristics**

  resource constraint → precedence constraint
- Job shop scheduling is a special case of this problem

Rome laboratory Outage MANager

- Shut down of a nuclear power plant for maintenance
  - **Should be done very carefully!**
- 10 000 to 40 000 jobs
- 1 000 000 euros per day



ROMAN problem

- Given a set of jobs, precedence constraints and resource requirements, assign resources to jobs for specific periods to:
  - Minimize makespan
  - Ensure that all jobs can be done safely
- **Jobs**: refueling, repairs, plant modifications, maintenance

Approach

- Decision tree for sophisticated safety analysis
- *Constraint programming* based project scheduling system
  - Builds on basic algorithms to be studied
- See papers!